CLAIMS

1. A process for preventing the piracy of application programs resident on a server and remotely accessed across a computer network by a client system in a computer environment, comprising the steps of:

providing a network filesystem on said client;

wherein said network filesystem handles and forwards all requests from local processes on said client that are directed at application program files located on said server;

wherein said filesystem examines each of said requests, and either grants or denies each of said requests depending on whether the request is justifiable from a security perspective by using infomation that includes, but is not limited to: the nature of the originating process, the history of previous access by the process, and/or the section of the targeted file being requested;

providing a network redirector component of said network filesystem; and

wherein said network redirector component makes visible to said network filesystem, a path that represents the server where said application program files are stored.

2. The process of claim 1, wherein said network filesystem registers dispatch routines with the client operating system that handle common file operations such as open, read, write and close; wherein a dispatch routine examines a file request and decides whether to grant or deny said file request; and wherein if said file request is granted then said dispatch routine forwards said file request to said server and sends back said server's response to said client operating system.

3. The process of claim 1, wherein when a local process on said client makes a file request for an application program file on said server, said client operating system calls one of said dispatch routines with said file request.

4. A process for preventing the piracy of application programs resident on a server and remotely accessed across a computer network by a client system in a computer environment, comprising the steps of:

providing a network filesystem on said client;

wherein said network filesystem determines the identity of the process that originates a relevant open, read, or write request for an application program file on said server;

wherein said network filesystem registers a callback routine with the client operating system that is invoked whenever a new process is created;

wherein said callback routine receives from said client operating system the pathname to the new process' executable and the new process' unique process ID;

wherein said callback routine stores said pathname to the new process' executable and said new process' unique process ID in a process data structure;

wherein said process data structure is consulted by said network filesystem while servicing a file request in order to match the process ID that originated the request with the pathname of the process' executable;

wherein said network filesystem extracts the root of the pathname of said process' executable, said pathname root uniquely identifies the storage device or remote server that provides said executable; and

wherein if said pathname root specifies a server that is known to be secure, as opposed to a local storage device that is insecure, then said file request is safe and is granted by said network filesystem, otherwise, said file request is denied.

5. A process for preventing the piracy of application programs resident on a server and remotely accessed across a computer network by a client system in a computer environment, comprising the steps of:

providing a network filesystem on said client;

wherein said network filesystem handles and forwards all requests from local processes on said client that are directed at application program files located on said server;

wherein said network filesystem detects when a remotely served executable file is being opened;

wherein said network filesystem determines the offset and length of said executable file's code section and stores said offset and length;

wherein said network filesystem checks if a read or write request is for a remote executable on said server;

wherein if said read or write request is for a remote executable, then the offset and length of the code section of said remote executable is retrieved from the data stored by said network filesystem and compared to the offset and length of said read or write request; and

wherein said read or write request is denied if said offset and length of said read or write request intersects with said offset and length of the code section of said remote executable.

6. A process for preventing the piracy of application programs resident on a server and remotely accessed across a computer network by a client system in a computer environment, comprising the steps of:

providing a network filesystem on said client;

wherein said network filesystem handles and forwards all requests from local processes on said client that are directed at application program files located on said server;

wherein said client contains a virtual memory subsystem;

wherein said network filesystem checks for the presence of the paging I/O flag upon receiving a read request;

wherein if said I/O flag is not present, then said read request did not come from said client system's virtual memory system and said read request is denied by said network filesystem; and

wherein if said I/O flag is present, the said read request originated from said client's virtual memory subsystem and said read request is granted by said network filesystem.

7. A process for preventing the piracy of application programs resident on a server and remotely accessed across a computer network by a client system in a computer environment, comprising the steps of:

providing a network filesystem on said client system;

wherein said network filesystem handles and forwards all requests from local processes on said client that are directed at application program files located on said server;

wherein said client system contains a virtual memory subsystem;

wherein said network filesystem examines said client system's program stack upon receiving a read request for a virtual memory page that causes a page fault in said client system's virtual memory subsystem;

wherein said client system's program stack holds information about the current state of said client system's processor;

wherein said network filesystem examines the execution pointer register stored in said client system's program stack; and

wherein said network filesystem grants said read request if said execution pointer is a memory address within the boundary of said virtual memory page.

8. A process for preventing the piracy of application programs resident on a server and remotely accessed across a computer network by a client system in a computer environment, comprising the steps of:

providing a network filesystem on said client system;

wherein said network filesystem handles and forwards all requests from local processes on said client that are directed at application program files located on said server;

wherein said network filesystem registers a callback routine with the client operating system that is invoked whenever a new process is created;

wherein said callback routine receives from said client operating system the new process' unique process ID;

wherein said callback routine creates an access history for said new process and records said new process' unique process ID in said access history;

wherein said network filesystem, upon receiving a read request for a program file served by said server, determines the process ID of the requesting process and makes an entry into said requesting process' access history and records the file name, offset, and length of the request made by said process for said program file;

wherein said network filesystem examines entries in said requesting process' access history that refer to said program file to determine if the pattern of accesses more closely resembles an attempted file copy than code execution;

wherein if said pattern of accesses resembles an attempted file copy then said network filesystem denies said read request; and

wherein if said pattern of accesses resembles code execution then said network filesystem grants said read request.

9. The process of claim 8, wherein said network filesystem determines if an access pattern resembles an attempted file copy by checking if the past predetermined number of read requests to a particular file have been sequential.

10. A program storage medium readable by a computer, tangibly embodying a program of instructions executable by the computer to perform method steps for preventing the piracy of application programs resident on a server and remotely accessed across a computer network by a client system in a computer environment, comprising the steps of:

providing a network filesystem on said client;

wherein said network filesystem handles and forwards all requests from local processes on said client that are directed at application program files located on said server;

wherein said filesystem examines each of said requests, and either grants or denies each of said requests depending on whether the request is justifiable from a security perspective by using information that includes, but is not limited to: the nature of the originating process, the history of previous access by the process, and/or the section of the targeted file being requested;

providing a network redirector component of said network filesystem; and

wherein said network redirector component makes visible to said network filesystem, a path that represents the server where said application program files are stored.

11. The method of claim 10, wherein said network filesystem registers dispatch routines with the client operating system that handle common file operations such as open, read, write and close; wherein a dispatch routine examines a file request and decides whether to grant or deny said file request; and wherein if said file request is granted then said dispatch routine forwards said file request to said server and sends back said server's response to said client operating system.

12. The method of claim 10, wherein when a local process on said client makes a file request for an application program file on said server, said client operating system calls one of said dispatch routines with said file request.

13. A program storage medium readable by a computer, tangibly embodying a program of instructions executable by the computer to perform method steps for preventing the piracy of application programs resident on a server and remotely accessed across a computer network by a client system in a computer environment, comprising the steps of:

providing a network filesystem on said client;

wherein said network filesystem determines the identity of the process that originates a relevant open, read, or write request for an application program file on said server;

wherein said network filesystem registers a callback routine with the client operating system that is invoked whenever a new process is created;

wherein said callback routine receives from said client operating system the pathname to the new process' executable and the new process' unique process ID;

wherein said callback routine stores said pathname to the new process' executable and said new process' unique process ID in a process data structure;

wherein said process data structure is consulted by said network filesystem while servicing a file request in order to match the process ID that originated the request with the pathname of the process' executable;

wherein said network filesystem extracts the root of the pathname of said process' executable, said pathname root uniquely identifies the storage device or remote server that provides said executable; and

wherein if said pathname root specifies a server that is known to be secure, as opposed to a local storage device that is insecure, then said file request is safe and is granted by said network filesystem, otherwise, said file request is denied.

14. A program storage medium readable by a computer, tangibly embodying a program of instructions executable by the computer to perform method steps for preventing the piracy of application programs resident on a server and remotely accessed across a computer network by a client system in a computer environment, comprising the steps of:

providing a network filesystem on said client;

wherein said network filesystem handles and forwards all requests from local processes on said client that are directed at application program files located on said server;

wherein said network filesystem detects when a remotely served executable file is being opened;

wherein said network filesystem determines the offset and length of said executable file's code section and stores said offset and length;

wherein said network filesystem checks if a read or write request is for a remote executable on said server;

wherein if said read or write request is for a remote executable, then the offset and length of the code section of said remote executable is retrieved from the data stored by said network filesystem and compared to the offset and length of said read or write request; and

wherein said read or write request is denied if said offset and length of said read or write request intersects with said offset and length of the code section of said remote executable.

15. A program storage medium readable by a computer, tangibly embodying a program of instructions executable by the computer to perform method steps for preventing the piracy of application programs resident on a server and remotely accessed across a computer network by a client system in a computer environment, comprising the steps of:

providing a network filesystem on said client;

wherein said network filesystem handles and forwards all requests from local processes on said client that are directed at application program files located on said server;

wherein said client contains a virtual memory subsystem;

wherein said network filesystem checks for the presence of the paging I/O flag upon receiving a read request;

wherein if said I/O flag is not present, then said read request did not come from said client system's virtual memory system and said read request is denied by said network filesystem; and

wherein if said I/O flag is present, the said read request originated from said client's virtual memory subsystem and said read request is granted by said network filesystem.

16. A program storage medium readable by a computer, tangibly embodying a program of instructions executable by the computer to perform method steps for preventing the piracy of application programs resident on a server and remotely accessed across a computer network by a client system in a computer environment, comprising the steps of:

providing a network filesystem on said client system;

wherein said network filesystem handles and forwards all requests from local processes on said client that are directed at application program files located on said server;

wherein said client system contains a virtual memory subsystem;

wherein said network filesystem examines said client system's program stack upon receiving a read request for a virtual memory page that causes a page fault in said client system's virtual memory subsystem;

wherein said client system's program stack holds information about the current state of said client system's processor;

wherein said network filesystem examines the execution pointer register stored in said client system's program stack; and

wherein said network filesystem grants said read request if said execution pointer is a memory address within the boundary of said virtual memory page.

17. A program storage medium readable by a computer, tangibly embodying a program of instructions executable by the computer to perform method steps for preventing the piracy of application programs resident on a server and remotely accessed across a computer network by a client system in a computer environment, comprising the steps of:

providing a network filesystem on said client system;

wherein said network filesystem handles and forwards all requests from local processes on said client that are directed at application program files located on said server;

wherein said network filesystem registers a callback routine with the client operating system that is invoked whenever a new process is created;

wherein said callback routine receives from said client operating system the new process' unique process ID;

wherein said callback routine creates an access history for said new process and records said new process' unique process ID in said access history;

wherein said network filesystem, upon receiving a read request for a program file served by said server, determines the process ID of the requesting process and makes an entry into said requesting process' access history and records the file name, offset, and length of the request made by said process for said program file;

wherein said network filesystem examines entries in said requesting process' access history that refer to said program file to determine if the pattern of accesses more closely resembles an attempted file copy than code execution;

wherein if said pattern of accesses resembles an attempted file copy then said network filesystem denies said read request; and

wherein if said pattern of accesses resembles code execution then said network filesystem grants said read request.

18. The method of claim 17, wherein said network filesystem determines if an access pattern resembles an attempted file copy by checking if the past predetermined number of read requests to a particular file have been sequential.

19. A process for preventing the piracy of application programs resident on a client system in a computer environment, comprising the steps of:

providing a filesystem on said client;

wherein said filesystem handles and forwards all file requests from local processes on said client;

wherein said filesystem examines each of said requests, and either grants or denies each of said requests depending on whether the request is justifiable from a security perspective by using information that includes, but is not limited to: the nature of the originating process, the history of previous access by the process, and/or the section of the targeted file being requested;

wherein said filesystem registers dispatch routines with the client operating system that handle common file operations such as open, read, write and close;

wherein a dispatch routine examines a file request and decides whether to grant or deny said file request; and

wherein if said file request is granted, then said dispatch routine allows the requested operation to proceed.

20. A process for preventing the piracy of application programs resident on a client system in a computer environment, comprising the steps of:

providing a filesystem on said client;

wherein said filesystem handles and forwards all file requests from local processes on said client;

wherein said filesystem detects when an executable file is being opened;

wherein said filesystem determines the offset and length of said executable file's code section and stores said offset and length;

wherein said filesystem checks if a read or write request is for an executable;

wherein if said read or write request is for an executable, then the offset and length of the code section of said executable is retrieved from the data stored by said filesystem and compared to the offset and length of said read or write request; and

wherein said read or write request is denied if said offset and length of said read or write request intersects with said offset and length of the code section of said executable.

21. A process for preventing the piracy of application programs resident on a client system in a computer environment, comprising the steps of:

providing a filesystem on said client;

wherein said filesystem handles and forwards all file requests from local processes on said client;

wherein said client contains a virtual memory subsystem;

wherein said filesystem checks for the presence of the paging I/O flag upon receiving a read request;

wherein if said I/O flag is not present, then said read request did not come from said client system's virtual memory system and said read request is denied by said filesystem; and

wherein if said I/O flag is present, the said read request originated from said client's virtual memory subsystem and said read request is granted by said filesystem.

22. A process for preventing the piracy of application programs resident on a client system in a computer environment, comprising the steps of:

providing a filesystem on said client system;

wherein said filesystem handles and forwards all file requests from local processes on said client;

wherein said client system contains a virtual memory subsystem;

wherein said filesystem examines said client system's program stack upon receiving a read request for a virtual memory page that causes a page fault in said client system's virtual memory subsystem;

wherein said client system's program stack holds information about the current state of said client system's processor;

wherein said filesystem examines the execution pointer register stored in said client system's program stack; and

wherein said filesystem grants said read request if said execution pointer is a memory address within the boundary of said virtual memory page.

23. A process for preventing the piracy of application programs resident on a client system in a computer environment, comprising the steps of:

providing a filesystem on said client system;

wherein said filesystem handles and forwards all file requests from local processes on said client;

wherein said filesystem registers a callback routine with the client operating system that is invoked whenever a new process is created;

wherein said callback routine receives from said client operating system the new process' unique process ID;

wherein said callback routine creates an access history for said new process and records said new process' unique process ID in said access history;

wherein said filesystem, upon receiving a read request, determines the process ID of the requesting process and makes an entry into said requesting process' access history and records the file name, offset, and length of the request made by said process for said program file;

wherein said filesystem examines entries in said requesting process' access history that refer to said program file to determine if the pattern of accesses more closely resembles an attempted file copy than code execution;

wherein if said pattern of accesses resembles an attempted file copy then said filesystem denies said read request; and

wherein if said pattern of accesses resembles code execution then said filesystem grants said read request.

24. The process of claim 23, wherein said filesystem determines if an access pattern resembles an attempted file copy by checking if the past predetermined number of read requests to a particular file have been sequential.

25. A program storage medium readable by a computer, tangibly embodying a program of instructions executable by the computer to perform method steps for preventing the piracy of application programs resident on a client system in a computer environment, comprising the steps of:

providing a filesystem on said client;

wherein said filesystem handles and forwards all file requests from local processes on said client;

wherein said filesystem examines each of said requests, and either grants or denies each of said requests depending on whether the request is justifiable from a security perspective by using information that includes, but is not limited to: the nature of the originating process, the history of previous access by the process, and/or the section of the targeted file being requested;

wherein said filesystem registers dispatch routines with the client operating system that handle common file operations such as open, read, write and close;

wherein a dispatch routine examines a file request and decides whether to grant or deny said file request; and

wherein if said file request is granted, then said dispatch routine allows the requested operation to proceed.

26. A program storage medium readable by a computer, tangibly embodying a program of instructions executable by the computer to perform method steps for preventing the piracy of application programs resident on a client system in a computer environment, comprising the steps of:

providing a filesystem on said client;

wherein said filesystem handles and forwards all file requests from local processes on said client;

wherein said filesystem detects when an executable file is being opened;

wherein said filesystem determines the offset and length of said executable file's code section and stores said offset and length;

wherein said filesystem checks if a read or write request is for an executable;

wherein if said read or write request is for an executable, then the offset and length of the code section of said executable is retrieved from the data stored by said filesystem and compared to the offset and length of said read or write request; and wherein said read or write request is denied if said offset and length of said read or write request intersects with said offset and length of the code section of said executable.

27. A program storage medium readable by a computer, tangibly embodying a program of instructions executable by the computer to perform method steps for preventing the piracy of application programs resident on a client system in a computer environment, comprising the steps of:

providing a filesystem on said client;

wherein said filesystem handles and forwards all file requests from local processes on said client;

wherein said client contains a virtual memory subsystem;

wherein said filesystem checks for the presence of the paging I/O flag upon receiving a read request;

wherein if said I/O flag is not present, then said read request did not come from said client system's virtual memory system and said read request is denied by said filesystem; and

wherein if said I/O flag is present, the said read request originated from said client's virtual memory subsystem and said read request is granted by said filesystem.

28. A program storage medium readable by a computer, tangibly embodying a program of instructions executable by the computer to perform method steps for preventing the piracy of application programs resident on a client system in a computer environment, comprising the steps of:

providing a filesystem on said client system;

wherein said filesystem handles and forwards all file requests from local processes on said client;

wherein said client system contains a virtual memory subsystem;

wherein said filesystem examines said client system's program stack upon receiving a read request for a virtual memory page that causes a page fault in said client system's virtual memory subsystem;

wherein said client system's program stack holds information about the current state of said client system's processor;

wherein said filesystem examines the execution pointer register stored in said client system's program stack; and

wherein said filesystem grants said read request if said execution pointer is a memory address within the boundary of said virtual memory page.

29. A program storage medium readable by a computer, tangibly embodying a program of instructions executable by the computer to perform method steps for preventing the piracy of application programs resident on a client system in a computer environment, comprising the steps of:

providing a filesystem on said client system;

wherein said filesystem handles and forwards all file requests from local processes on said client;

wherein said filesystem registers a callback routine with the client operating system that is invoked whenever a new process is created;

wherein said callback routine receives from said client operating system the new process' unique process ID;

wherein said callback routine creates an access history for said new process and records said new process' unique process ID in said access history;

wherein said filesystem, upon receiving a read request, determines the process ID of the requesting process and makes an entry into said requesting process' access history and records the file name, offset, and length of the request made by said process for said program file;

wherein said filesystem examines entries in said requesting process' access history that refer to said program file to determine if the pattern of accesses more closely resembles an attempted file copy than code execution;

wherein if said pattern of accesses resembles an attempted file copy then said filesystem denies said read request; and

wherein if said pattern of accesses resembles code execution then said filesystem grants said read request.

30. The method of claim 29, wherein said filesystem determines if an access pattern resembles an attempted file copy by checking if the past predetermined number of read requests to a particular file have been sequential.